

## Lab 14 – Criando GUI com Swing

Neste laboratório irá desenvolver uma interface gráfica com eventos que será construída ao longo deste laboratório.

Sugerimos que estes exemplos sejam feitos com uso da IDE Eclipse.

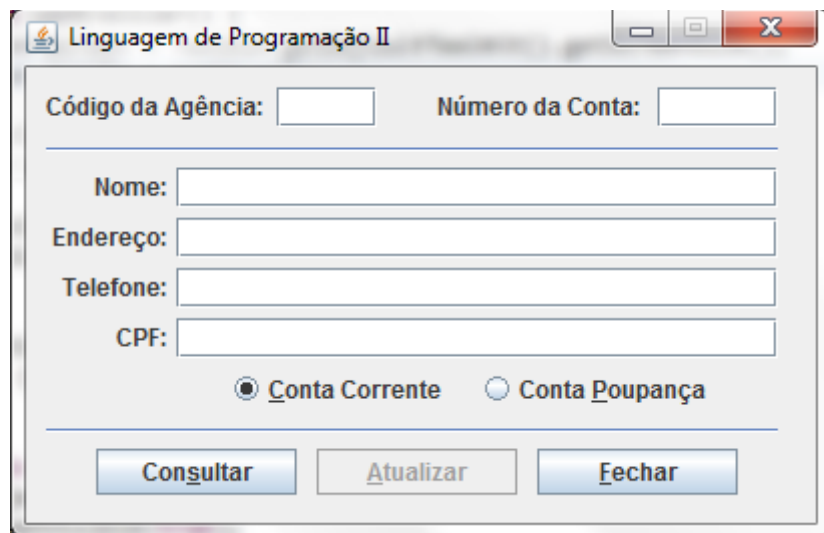
Antes faça a instalação do Window Builder no eclipse para trabalhar interfaces GUI Swing.

Siga o tutorial [Tutorial Window Builder](#)

**Duração prevista: 40 minutos**

### Exercícios

**Exercício 1:** Desenvolver a janela abaixo. (30 minutos)



**Exercício 2:** Manipulação de eventos com Java Swing (10 minutos)

### Exercício 1 - Desenvolver janela swing.

1. Crie uma classe chamada Janela que herde as características da superclasse javax.swing.JFrame. Essa janela deve ter as seguintes características:

```
public final class Janela extends JFrame { }
```

- a. O tamanho da Janela deve ser de 400 x 255 pixels
- b. O título da Janela deve ser "Linguagem de Programação II"

```
public Janela() {  
    setSize(400, 255);  
    setTitle("Linguagem de Programação II");  
}
```

- c. Implemente o método `centralizar()`, para centralizar a janela na área de trabalho do usuário. Esse método deve ser chamado dentro do construtor da classe `Janela`

```
private void centralizar() {
    Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension janela = getSize();

    if (janela.height > screen.height) {
        setSize(janela.width, screen.height);
    }

    if (janela.width > screen.width) {
        setSize(screen.width, janela.height);
    }

    setLocation((screen.width - janela.width) / 2,
        (screen.height - janela.height) / 2);
}
```

- d. Para evitar que o usuário redimensione a janela, adicione o código abaixo no construtor da classe `Janela`:

```
setResizable(false);
```

- e. Para poder ajustar os componentes livremente na janela, você deve definir o gerenciador de Layouts do container `JFrame` como nulo. Para isso, adicione o código abaixo dentro do construtor da classe `Janela`:

```
getContentPane().setLayout(null);
```

- f. Para evitar que a aplicação continue executando após o usuário clicar no botão fechar da janela, adicione o código abaixo dentro do construtor da classe `Janela`:

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

2. Para executar a classe `Janela`, adicione um método `main()` com o seguinte código:

```
public static void main(String args[]) {
    Janela janela = new Janela();
    janela.setVisible(true);
}
```

3. Crie os seguintes atributos na classe `Janela`:

- `jIAgencia` (`javax.swing.JLabel`): Rótulo do campo agência
- `jtfAgencia` (`javax.swing.JTextField`): Campo de texto para digitar o número da agência
- `jIConta` (`javax.swing.JLabel`): Rótulo do campo conta
- `jtfConta` (`javax.swing.JTextField`): Campo de texto para digitar o número da conta
- `jSeparator01` (`javax.swing.JSeparator`): Separador que vamos utilizar para separar as informações bancárias das informações do cliente
- `jINome` (`javax.swing.JLabel`): Rótulo do campo nome
- `jtfNome` (`javax.swing.JTextField`): Campo de texto para digitar o nome do cliente
- `jIEndereco` (`javax.swing.JLabel`): Rótulo do campo endereço
- `jtfEndereco` (`javax.swing.JTextField`): Campo de texto para digitar o endereço do cliente
- `jITelefone` (`javax.swing.JLabel`): Rótulo do campo telefone
- `jtfTelefone` (`javax.swing.JTextField`): Campo de texto para digitar o telefone do cliente
- `jICpf` (`javax.swing.JLabel`): Rótulo do campo CPF
- `jtfCpf` (`javax.swing.JTextField`): Campo de texto para digitar o CPF do cliente

- n. `jICorrente` (`javax.swing.JLabel`): Rótulo do campo Conta Corrente
- o. `jrbCorrente` (`javax.swing.JRadioButton`): Botão de rádio para selecionar contas do tipo “Conta Corrente”
- p. `jIPoupanca` (`javax.swing.JLabel`): Rótulo do campo Conta Poupanca
- q. `jrbPoupanca` (`javax.swing.JRadioButton`): Botão de rádio para selecionar contas do tipo “Conta Poupança”
- r. `bgContas` (`javax.swing.ButtonGroup`): Contêiner para agrupar os componentes do tipo `JRadioButton`
- s. `jSeparator02` (`javax.swing.JSeparator`): Separador que vamos utilizar para separar as informações do cliente do botões da janela
- t. `jbConsultar` (`javax.swing.JButton`): Botão utilizado para realizar uma consulta nas contas da agência bancária
- u. `jbAtualizar` (`javax.swing.JButton`): Botão utilizado para atualizar as informações da conta bancária
- v. `jbFechar` (`javax.swing.JButton`): Botão utilizado para fechar a aplicação

```
private JLabel jIAgencia;  
private JTextField jtfAgencia;  
private JLabel jIConta;  
private JTextField jtfConta;  
private JSeparator jSeparator01;  
private JLabel jINome;  
private JTextField jtfNome;  
private JLabel jIEndereco;  
private JTextField jtfEndereco;  
private JLabel jITelefone;  
private JTextField jtfTelefone;  
private JLabel jICpf;  
private JTextField jtfCpf;  
private JLabel jICorrente;  
private JRadioButton jrbCorrente;  
private JLabel jIPoupanca;  
private JRadioButton jrbPoupanca;  
private ButtonGroup bgContas;  
private JSeparator jSeparator02;  
private JButton jbConsultar;  
private JButton jbAtualizar;  
private JButton jbFechar;
```

- 4. Crie a instância do componente `jIAgencia` e configure as seguintes opções:
  - a. O texto do componente `jIAgencia` deve ser “Código da Agência:”
  - b. O tamanho do componente `jIAgencia` deve ser de 110 x 18 pixels.
  - c. O componente `jIAgencia` deve ser posicionado no pixel 10 x 10 da janela
  - d. Adicione o componente `jIAgencia` no container da Janela

```
jIAgencia = new JLabel("Código da Agência:");  
jIAgencia.setBounds(10, 10, 110, 18);  
add(jIAgencia);
```

- 5. Crie a instância do componente `jtfAgencia` e configure as seguintes opções:
  - a. O tamanho do componente `jtfAgencia` deve ser de 50 x 20 pixels
  - b. O componente `jtfAgencia` deve ser posicionado no pixel 125 x 10 da janela
  - c. Adicione o componente `jtfAgencia` no container da Janela

```
jtfAgencia = new JTextField();  
jtfAgencia.setBounds(125, 10, 50, 20);
```

```
add(jtfAgencia);
```

6. Crie a instância do componente `JlConta` e configure as seguintes opções:
  - a. O texto do componente `JlConta` deve ser "Número da Conta:"
  - b. O tamanho do componente `JlConta` deve ser de 105 x 18 pixels.
  - c. O componente `JlConta` deve ser posicionado no pixel 205 x 10 da janela
  - d. Adicione o componente `JlConta` no container da Janela
  
7. Crie a instância do componente `JtfConta` e configure as seguintes opções:
  - a. O tamanho do componente `JtfConta` deve ser de 60 x 20 pixels
  - b. O componente `JtfConta` deve ser posicionado no pixel 315 x 10 da janela
  - c. Adicione o componente `JtfConta` no container da Janela
  
8. Crie a instância do componente `JSeparator01` e configure as seguintes opções:
  - a. O tamanho do componente `JSeparator01` deve ser de 365 x 10 pixels
  - b. O componente `JSeparator01` deve ser posicionado no pixel 10 x 40 da janela
  - c. Adicione o componente `JSeparator01` no container da Janela

```
JSeparator01 = new JSeparator();  
JSeparator01.setBounds(10, 40, 365, 10);  
add(JSeparator01);
```
  
9. Crie a instância do componente `JlNome` e configure as seguintes opções:
  - a. O texto do componente `JlNome` deve ser "Nome:"
  - b. O tamanho do componente `JlNome` deve ser de 60 x 18 pixels.
  - c. O componente `JlNome` deve ser posicionado no pixel 10 x 50 da janela
  - d. Alinhe o texto do componente à direita.  
`JlNome.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);`
  - e. Adicione o componente `JlNome` no container da Janela
  
10. Crie a instância do componente `JtfNome` e configure as seguintes opções:
  - a. O tamanho do componente `JtfNome` deve ser de 300 x 20 pixels
  - b. O componente `JtfNome` deve ser posicionado no pixel 75 x 50 da janela
  - c. Adicione o componente `JtfNome` no container da Janela
  
11. Crie a instância do componente `JlEndereco` e configure as seguintes opções:
  - a. O texto do componente `JlEndereco` deve ser "Endereço:"
  - b. O tamanho do componente `JlEndereco` deve ser de 60 x 18 pixels.
  - c. O componente `JlEndereco` deve ser posicionado no pixel 10 x 75 da janela
  - d. Alinhe o texto do componente `JlEndereco` à direita.
  - e. Adicione o componente `JlEndereco` no container da Janela

12. Crie a instância do componente `jtfEndereco` e configure as seguintes opções:
- O tamanho do componente `jtfEndereco` deve ser de 300 x 20 pixels
  - O componente `jtfEndereco` deve ser posicionado no pixel 75 x 75 da janela
  - Adicione o componente `jtfEndereco` no container da Janela
13. Crie a instância do componente `jlTelefone` e configure as seguintes opções:
- O texto do componente `jlTelefone` deve ser "Telefone:"
  - O tamanho do componente `jlTelefone` deve ser de 60 x 18 pixels.
  - O componente `jlTelefone` deve ser posicionado no pixel 10 x 100 da janela
  - Alinhe o texto do componente `jlTelefone` à direita.
  - Adicione o componente `jlTelefone` no container da Janela
14. Crie a instância do componente `jtfTelefone` e configure as seguintes opções:
- O tamanho do componente `jtfTelefone` deve ser de 300 x 20 pixels
  - O componente `jtfTelefone` deve ser posicionado no pixel 75 x 100 da janela
  - Adicione o componente `jtfTelefone` no container da Janela
15. Crie a instância do componente `jlCpf` e configure as seguintes opções:
- O texto do componente `jlCpf` deve ser "CPF:"
  - O tamanho do componente `jlCpf` deve ser de 60 x 18 pixels.
  - O componente `jlCpf` deve ser posicionado no pixel 10 x 125 da janela
  - Alinhe o texto do componente `jlCpf` à direita.
  - Adicione o componente `jlCpf` no container da Janela
16. Crie a instância do componente `jtfCpf` e configure as seguintes opções:
- O tamanho do componente `jtfCpf` deve ser de 300 x 20 pixels
  - O componente `jtfCpf` deve ser posicionado no pixel 75 x 125 da janela
  - Adicione o componente `jtfCpf` no container da Janela
17. Crie a instância do componente `jlCorrente` e configure as seguintes opções:
- O texto do componente `jlCorrente` deve ser "Conta Corrente:"
  - O tamanho do componente `jlCorrente` deve ser de 100 x 18 pixels.
  - O componente `jlCorrente` deve ser posicionado no pixel 120 x 150 da janela
  - Adicione o componente `jlCorrente` no container da Janela
18. Crie a instância do componente `jrbCorrente` e configure as seguintes opções:
- O tamanho do componente `jrbCorrente` deve ser de 111 x 20 pixels
  - O componente `jrbCorrente` deve ser posicionado no pixel 100 x 150 da janela
  - Configure o atalho (alt + c) para o componente `jrbCorrente`  
`jrbCorrente.setMnemonic('C');`

- d. Por padrão, o radio da Conta Corrente estará selecionado quando o usuário abrir a janela. Para isso, adicione o código abaixo:

```
jrbCorrente.setSelected(true);
```

- e. Adicione o componente jrbCorrente no container da Janela

19. Crie a instância do componente jlPoupanca e configure as seguintes opções:

- O texto do componente jlPoupanca deve ser "Conta Corrente:"
- O tamanho do componente jlPoupanca deve ser de 100 x 18 pixels.
- O componente jlPoupanca deve ser posicionado no pixel 245 x 150 da janela
- Adicione o componente jlPoupanca no container da Janela

20. Crie a instância do componente jrbPoupanca e configure as seguintes opções:

- O tamanho do componente jrbPoupanca deve ser de 118 x 20 pixels
- O componente jrbPoupanca deve ser posicionado no pixel 225 x 150 da janela
- Configure o atalho (alt + p) para o componente jrbPoupanca

```
jrbPoupanca.setMnemonic('P');
```

- Adicione o componente jrbPoupanca no container da Janela

21. Para garantir que apenas um botão radio seja selecionado pelo usuário, temos que agrupar os componentes jrbCorrente e jrbPoupanca em um container do tipo ButtonGroup. Crie a instância do container bgContas e adicione os componentes jrbCorrente e jrbPoupanca nele.

```
bgContas = new ButtonGroup();  
bgContas.add(jrbCorrente);  
bgContas.add(jrbPoupanca);
```

22. Crie a instância do componente jSeparator02 e configure as seguintes opções:

- O tamanho do componente jSeparator02 deve ser de 365 x 10 pixels
- O componente jSeparator02 deve ser posicionado no pixel 10 x 180 da janela
- Adicione o componente jSeparator02 no container da Janela

23. Crie a instância do componente jbConsultar e configure as seguintes opções:

- O tamanho do componente jbConsultar deve ser de 100 x 23 pixels
- O componente jbConsultar deve ser posicionado no pixel 35 x 190 da janela
- Configure o atalho (alt + s) para o componente jbConsultar
- Adicione o componente jbConsultar no container da Janela

```
jbConsultar = new JButton("Consultar");  
jbConsultar.setBounds(35, 190, 100, 23);  
jbConsultar.setMnemonic('S');  
add(jbConsultar);
```

24. Crie a instância do componente jbAtualizar e configure as seguintes opções:

- O tamanho do componente jbAtualizar deve ser de 100 x 23 pixels
- O componente jbAtualizar deve ser posicionado no pixel 145 x 190 da janela

- c. Configure o atalho (alt + a) para o componente jbAtualizar
- d. Por padrão, o componente jbAtualizar deve ficar desabilitado. Para isso, adicione o código abaixo:

```
jbAtualizar.setEnabled(false);
```

- e. Adicione o componente jbAtualizar no container da Janela

25. Crie a instância do componente jbFechar e configure as seguintes opções:

- a. O tamanho do componente jbFechar deve ser de 100 x 23 pixels
- b. O componente jbFechar deve ser posicionado no pixel 225 x 190 da janela
- c. Configure o atalho (alt + f) para o componente jbFechar
- d. Adicione o componente jbFechar no container da Janela

## ***Exercício 2 - Manipulação de eventos com Java Swing.***

1. Utilizando a janela implementada no exercício anterior crie os seguintes eventos.

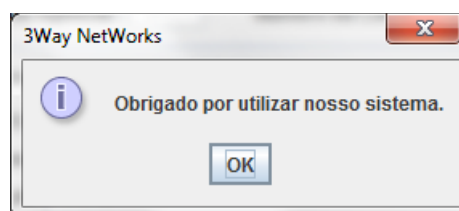
1.1. Utilize o método `windowOpened()` da interface `WindowListener`, para exibir a caixa de diálogo abaixo no momento em que o usuário abrir a aplicação.

- a. Dentro do construtor da classe Janela, faça uma chamada ao método `addWindowListener()`. Esse método adiciona um "ouvinte" a sua Janela.
- b. Dentro do parâmetro do método `addWindowListener()` crie uma classe do tipo `java.awt.event.WindowAdapter`
- c. Dentro da classe `WindowAdapter`, sobrescreva o método `windowOpened()` da interface `WindowListener`
- d. Dentro desse método, faça uma chamada para qualquer método da classe Janela.
- e. Dentro desse método, você deve criar uma caixa de diálogo (`JOptionPane`) para exibir a mensagem na tela. Veja o exemplo abaixo:

```
addWindowListener(new WindowAdapter() {  
    @Override  
    public void windowOpened(WindowEvent evt) {  
        JOptionPane.showMessageDialog(null, "Programação Java OO", "3Way NetWorks",  
        JOptionPane.INFORMATION_MESSAGE);  
    }  
});
```

1.2. Utilize o método `windowClosing()` da interface `WindowListener`, para exibir a caixa de diálogo abaixo no momento em que o usuário fechar a aplicação.

- a. Dentro da classe `WindowAdapter` (criada anteriormente), sobrescreva o método `windowClosing()` da interface `WindowListener`



- b. Dentro desse método, faça uma chamada para qualquer método da classe Janela onde será exibida a caixa de diálogo acima.

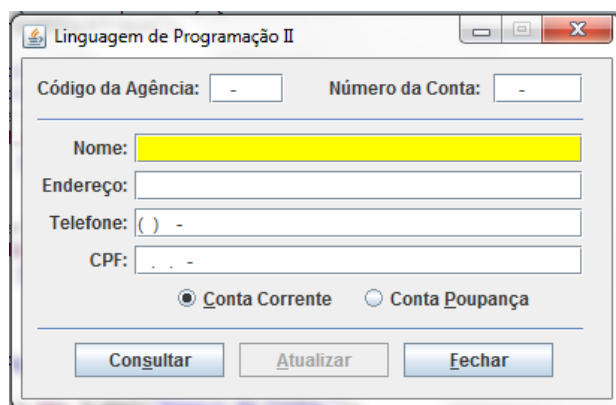
1.3. A janela contém seis caixas de textos (`jtfAgencia`, `jtfConta`, `jtfNome`, `jtfEndereco`, `jtfTelefone` e `jtfCpf`). No momento em que o usuário ativar algum desses componentes, a cor do componente

selecionado será alterado para amarelo. No momento em que o usuário sair do componente, a cor deve voltar a ser branca. Para isso, implemente a lógica abaixo em todos os componentes citados anteriormente.

- Dentro do construtor da classe `Janela`, após definir o tamanho e a posição de cada `TextField`, faça uma chamada ao método `addFocusListener()` a partir do componente criado. Esse método adiciona um "ouvinte" que gerencia os focus do seu componente.
- Dentro do parâmetro do método `addFocusListener()` crie uma classe do tipo `java.awt.event.FocusAdapter`
- Dentro da classe `FocusAdapter`, sobrescreva o método `focusGained()` da interface `FocusListener`
- Dentro desse método, faça uma chamada para qualquer método da classe `Janela`. Veja o exemplo abaixo:
- Dentro desse método, você deve alterar a cor do componente selecionado pelo usuário:
- Essa mesma lógica deve ser utilizada para gerenciar quando o componente perde o focus. Dentro da classe `FocusAdapter` (criada anteriormente), sobrescreva o método `focusLost()` da interface `FocusListener`
- Dentro desse método, faça uma chamada para qualquer método da classe `Janela` onde será alterada a cor do componente.
- O resultado dessa implementação deve ser parecida com o exemplo abaixo:

```
jtfAgencia.addFocusListener(new FocusAdapter() {
    @Override
    public void focusGained(FocusEvent evt) {
        jtfAgencia.setBackground(Color.YELLOW);
    }

    @Override
    public void focusLost(FocusEvent evt) {
        jtfAgencia.setBackground(Color.WHITE);
    }
});
```



1.4. As caixas de textos `jtfAgencia`, `jtfConta`, `jtfTelefone` e `jtfCpf` são todas numéricas. Desse modo, adicione aos componentes uma máscara para permitir que somente caracteres numéricos sejam digitados pelo usuário. Já que estamos trabalhando com máscaras, vamos formatar o texto digitado pelo usuário.

- A classe que controla as máscaras das caixas de texto é `javax.swing.text.MaskFormatter`. A classe `MaskFormatter` contém um construtor alternativo que recebe uma `String` com a máscara formatada. Por exemplo, para criar uma máscara para o CPF devemos criar o objeto com a `String`: `"###.###.###-##"`
- A máscara não pode ser aplicada diretamente em objetos do tipo `TextField`. Mantenha a declaração das caixas de texto como `TextField`, porém as instâncias criadas para esses



objetos serão do tipo JFormattedTextField (polimorfismo).

- O componente jtfAgencia deve ter a seguinte máscara: "####-#"
- O componente jtfConta deve ter a seguinte máscara: "#####-#"
- O componente jtfTelefone deve ter a seguinte máscara: "(0xx##) ####-####"
- O componente jtfCpf deve ter a seguinte máscara: "###.###.###-##"
- O caractere # serve como um coringa, e só pode ser substituído por um caractere numérico
- Para utilizar a classe MaskFormatter é necessário fazer o tratamento de exceção do tipo ParseException na criação da instância. Veja um exemplo abaixo:

```
try {
    jtfAgencia = new JFormattedTextField(new MaskFormatter("####-#"));
} catch (ParseException e) {
    e.printStackTrace();
}
```

1.2. Finalizando essa lista de exercício, vamos adicionar os eventos para componentes do tipo JButton ao nosso projeto. Para isso:

- Utilize o método actionPerformed() da interface ActionListener, para controlar os eventos nos botões jbConsultar, jbAtualizar e jbFechar
- Dentro do construtor da classe Janela, após definir o tamanho, a posição e o atalho de cada JButton, faça uma chamada ao método addActionListener() a partir do componente criado. Esse método adiciona um "ouvinte" que gerencia todos os eventos que o usuário pode provocar nos botões a partir do teclado ou do mouse
- Dentro do parâmetro do método addActionListener() crie uma classe do tipo java.awt.event.ActionListener
- Dentro da classe ActionListener, implemente o método actionPerformed()
- Dentro desse método, faça uma chamada para qualquer método da classe Janela.
- Adicione um evento para o componente jbConsultar, que exiba a caixa de diálogo abaixo caso o usuário não preencha ao menos um dos campos jtfAgencia e jtfConta. Veja o exemplo abaixo:

```
jbConsultar.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        if(jtfAgencia.getText().equals(" - ") || jtfConta.getText().equals(" - ")){
            JOptionPane.showMessageDialog(null, "E' necessário informar a agência e a conta", "3Way
NetWorks", JOptionPane.INFORMATION_MESSAGE);
        }

    }
});
```

- Adicione um evento para o componente jbFechar, que feche a aplicação caso o usuário clique no botão Fechar. Para fechar a aplicação, utilize o comando abaixo:

```
jbFechar.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
});
```